The BICOR and CORS iterative algorithms for solving nonsymmetric linear systems

Carpentieri, B.; Jing, Y. -F.; Huang, T. -Z.

Published in:
SIAM: Journal on Scientific Computing

DOI:
10.1137/100794031

Publication date:
2011

SIAM J. SCI. COMPUT.
Vol. 33, No. 5, pp. 3020–3036

THE BICOR AND CORS ITERATIVE ALGORITHMS FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS*

B. CARPENTIERI†, Y.-F. JING‡, AND T.-Z. HUANG‡

Abstract. We present two iterative algorithms for solving real nonsymmetric and complex non-Hermitian linear systems of equations and that were developed from variants of the nonsymmetric Lanczos method. In this paper, we give the theoretical background of the two iterative methods and discuss their main computational aspects. Using a large number of numerical experiments, we analyze their convergence properties, and we also compare them with other popular nonsymmetric iterative solvers in use today.

Key words. Krylov subspace methods, linear systems, nonsymmetric Lanczos method, sparse and dense matrix computation

AMS subject classifications. 65F10, 65F50, 65N22, 65H10

DOI. 10.1137/100794031

1. Introduction.

In this paper we discuss iterative solutions of large sparse and/or dense linear systems of equations

$$Ax = b, \tag{1.1}$$

where the coefficient matrix A is nonsymmetric and possibly indefinite. One of the most popular nonsymmetric iterative procedures is the generalized minimum residual (GMRES) method proposed by Saad and Schultz [25]. The GMRES method computes the *optimal* approximation x_k for which the 2-norm of the residual is minimal over the Krylov space $K_k(A, r_0) = span\left\{r_0, Ar_0, \ldots, A^{k-1}r_0\right\}$ for $k = 1, 2, \ldots$. The number of arithmetic operations required and the storage used to perform the kth GMRES iteration are $\mathcal{O}(nk)$. This means that the cost of applying the method increases with the iterations, and it may sometimes become prohibitively large for solving practical applications (see, e.g., [3]). As an attempt to limit the costs of GMRES while preserving its favorable convergence properties, most implementations restart the algorithm after a number of steps and, in some cases, augment the corresponding Krylov space with extra information, or keep only selected information from it (see, e.g., [21, 9]).

On the other hand, cost considerations have motivated (and continue to motivate) the development of *nonoptimal* methods built upon short-term vector recurrences, which require only $\mathcal{O}(n)$ extra storage in addition to the matrix and perform $\mathcal{O}(n)$ operations. The principal developments in the field of nonoptimal methods include the biconjugate gradient (BiCG) method developed by Fletcher [13], the conjugate gradient squared (CGS) method proposed by Sonneveld [30], Freund and Nachtigal's quasi-minimal residual (QMR) method [16] and transpose-free quasi-minimal residual method [15], van der Vorst's biconjugate gradient stabilized (BiCGSTAB)

*Received by the editors May 3, 2010; accepted for publication (in revised form) July 19, 2011; published electronically October 27, 2011.

http://www.siam.org/journals/sisc/33-5/79403.html

†Institute of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands (b.carpentieri@rug.nl).

‡School of Mathematical Sciences/Institute of Computational Science, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P. R. China (yanfeijing@uestc.edu.cn, tzhuang@uestc.edu.cn). The work of these authors was supported by NSFC 60973015, 973 Program (2007CB311002), and Sichuan Province Sci. & Tech. Research Project (2009SPT-1).

method [32], and the BiCGSTAB(l) method introduced by Sleijpen and Fokkema [26]. If the matrix-vector product is expensive, as in the case of A dense, GMRES may be more efficient because it minimizes the norm of the residual at each step, and thus it converges in the fewest iterations. If the storage is limited, or the matrix-vector product costs $\mathcal{O}(n)$, some of these nonoptimal algorithms may converge faster than GMRES in solution time, thanks to their limited expenses. See also general discussions in [17, 11].

It is a research question to determine the classes of problems for which one algorithm is more memory efficient than others. For example, BiCGSTAB and CGS often exhibit the best performance rates for solving some second-order elliptic partial differential equations with variable coefficients discretized on a regular Cartesian grid [8]. Sometimes QMR outperforms all of the other methods [6], while at other times GMRES is the most competitive, as on some convection-diffusion [35] and boundary integral equations [3]. Faber and Manteuffel showed that no Krylov subspace method for solving nonsymmetric linear systems can both be optimal and use short-term recurrences of fixed length [12]. Therefore, there is no alternative to this trend of development.

In our research, we explore variants of the Lanczos method for solving real nonsymmetric and/or complex non-Hermitian linear systems with the motivation of obtaining smoother and, we hope, faster convergence behavior in comparison with the BiCG method as well as its two evolving variants: the CGS method and, one of the most popular methods in use today, the BiCGSTAB method. We describe two iterative algorithms developed from the biconjugate A-orthonormalization procedure. The procedure is built upon the two-sided Lanczos method and can be seen as a Petrov–Galerkin projection technique between the dual Krylov subspaces $K_m(A; v_1)$ and $A^T K_m(A^T; w_1)$ in the real case, which is $A^H K_m(A^H; w_1)$ in the complex case, where v_1 and w_1 are two column vectors of appropriate dimension. The generation uses three-term vector recurrences in contrast with the Arnoldi algorithm for nonsymmetric matrices used in GMRES.

This paper is structured as follows. In section 2, we provide the theoretical background of the Petrov–Galerkin projection technique, and in section 3 we discuss its application to the solution of linear systems. In sections 4 and 5 we present the biconjugate A-orthogonal residual (BiCOR) and the conjugate A-orthogonal residual squared (CORS) methods. Finally, in section 6 we analyze the numerical behavior of the two methods using a large number of numerical experiments. This study integrates and extends a preliminary analysis reported in [20] for solving small complex non-Hermitian linear systems. The set of matrix problems is much wider than in [20] and includes both sparse and dense and real and complex problems arising in different areas of two orders of magnitude larger.

2. Two-sided biconjugate *A*-orthonormalization procedure for nonsymmetric matrices. Throughout this paper we denote by the superscript T the transpose of a vector or a matrix and the standard inner product of two real vectors $u, v \in \mathbb{R}^n$ by

$$\langle u, v \rangle = u^T v = \sum_{i=1}^{n} u_i v_i.$$

Given two vectors v_1 and w_1 with Euclidean inner product $\langle w_1, A v_1 \rangle = 1$, we define Lanczos-type vectors v_j, w_j and scalars δ_j, β_j, $j = 1, 2, \ldots$, by the recursions

$$\delta_{j+1} v_{j+1} = A v_j - \beta_j v_{j-1} - \alpha_j v_j, \tag{2.1}$$

$$\beta_{j+1} w_{j+1} = A^T w_j - \delta_j w_{j-1} - \alpha_j w_j, \tag{2.2}$$

where the scalars are chosen as

$$\alpha_j = \langle w_j, A^2 v_j \rangle, \ \beta_j = \langle w_{j-1}, A^2 v_j \rangle, \ \delta_j = \langle w_j, A^2 v_{j-1} \rangle.$$

This choice of the scalars guarantees that the recursions (2.1)–(2.2) generate sequences of biconjugate A-orthonormal vectors (or briefly, A-biorthonormal vectors) v_i and w_i, according to the following definition.

DEFINITION 1. *Right and left Lanczos-type vectors v_j, $j = 1, 2, \ldots, m$, and w_i, $i = 1, 2, \ldots, m$, form a biconjugate A-orthonormal system in exact arithmetic if and only if*

$$\langle w_i, A v_j \rangle = \delta_{i,j}, \ 1 \le i, j \le m.$$

The sequences may be derived from the standard two-sided Lanczos procedure for nonsymmetric matrices using suitable initial vectors v_1, w_1. Therefore, we may see (2.1)–(2.2) as a two-sided Gram–Schmidt orthonormalization procedure. At step i we multiply vectors v_i and w_i by A and A^T, respectively, and we orthonormalize them against the most recently generated Lanczos-type pairs of vectors (v_i, w_i) and (v_{i-1}, w_{i-1}). The vectors $\alpha_i v_i$, $\alpha_i w_i$ are the biconjugate A-orthonormal projections of Av_i and $A^T w_i$ onto the most recently computed vectors v_i and w_i; analogously, the vectors $\beta_i v_{i-1}$, $\delta_i w_{i-1}$ are the biconjugate A-orthonormalization projections of Av_i and $A^T w_i$ onto the previously computed vectors v_{i-1} and w_{i-1}. Notice that the scalars β_i and δ_i can be chosen with some freedom, provided the biconjugate A-orthonormalization property holds.

For clarity, we sketch a complete version of the biconjugate A-orthonormalization procedure in Algorithm 1. The storage is clearly very limited compared to the Arnoldi procedure. At step j, only the two most recently computed pairs of Lanczos-type vectors v_k and w_k for $k = j, j-1$ are needed to generate v_{j+1}, w_{j+1}. The two vectors may be overwritten with the most recent updates. The matrix A is not modified and is accessed only via matrix-vector products by A and A^T. The above algorithm can possibly fail if δ_{j+1} vanishes, while $\tilde{w}_{j+1}$ and $A\tilde{v}_{j+1}$ appearing in line 7 are not equal to $0 \in \mathbb{R}^n$. A real implementation would try to recover from a possible

ALGORITHM 1. *The biconjugate A-orthonormalization procedure.*

1: Choose v_1, w_1, such that $\langle w_1, A v_1 \rangle = 1$
2: Set $\beta_1 = \delta_1 \equiv 0, w_0 = v_0 = \mathbf{0} \in \mathbb{R}^n$
3: **for** $j = 1, 2, \ldots$ **do**
4: $\quad \alpha_j = \langle w_j, A(A v_j) \rangle$
5: $\quad \tilde{v}_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$
6: $\quad \tilde{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
7: $\quad \delta_{j+1} = |\langle \tilde{w}_{j+1}, A\tilde{v}_{j+1} \rangle|^{\frac{1}{2}}$
8: $\quad \beta_{j+1} = \dfrac{\langle \tilde{w}_{j+1}, A\tilde{v}_{j+1} \rangle}{\delta_{j+1}}$
9: $\quad v_{j+1} = \dfrac{\tilde{v}_{j+1}}{\delta_{j+1}}$
10: $\quad w_{j+1} = \dfrac{\tilde{w}_{j+1}}{\beta_{j+1}}$
11: **end for**

failure by using remedies such as the so-called look-ahead strategies [23, 22, 14, 18]. But this issue is outside the scope of this paper and we shall not pursue it here.

The recurrences (2.1)–(2.2) ideally build up a pair of biconjugate A-orthonormal bases for the dual Krylov subspaces $K_m(A; v_1)$ and $A^T K_m(A^T; w_1)$ as shown in the following proposition.

PROPOSITION 1. *If Algorithm* 1 *proceeds* m *steps, then the right and left Lanczos-type vectors* $v_j, j = 1, 2, \ldots, m$, *and* $w_i, i = 1, 2, \ldots, m$, *form a biconjugate* A*-orthonormal system in exact arithmetic, i.e.,*

$$\langle w_i, Av_j \rangle = \delta_{i,j}, \quad 1 \le i,\ j \le m.$$

Next denote by $V_m = [v_1, v_2, \ldots, v_m]$ *and* $W_m = [w_1, w_2, \ldots, w_m]$ *the* $n \times m$ *matrices, and denote by* $\underline{T_m}$ *the extended tridiagonal matrix of the form*

$$\underline{T_m} = \begin{bmatrix} T_m \\ \delta_{m+1} e_m^T \end{bmatrix}, \tag{2.3}$$

where

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{bmatrix},$$

whose entries are the coefficients generated during the algorithm implementation, and in which $\alpha_1, \ldots, \alpha_m, \beta_2, \ldots, \beta_m$ *are complex while* $\delta_2, \ldots, \delta_m$ *are positive. Then the following four relations hold:*

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T, \tag{2.4}$$

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T, \tag{2.5}$$

$$W_m^T A V_m = I_m, \tag{2.6}$$

$$W_m^T A^2 V_m = T_m, \tag{2.7}$$

where e_m *is the* m*-vector having* m*th component equal to one and all other components zero. The matrix* T_m *can be seen as the projection of the matrix* A *onto the corresponding Krylov subspaces.*

Proof. We show by induction that the right and left Lanczos-type vectors $v_j, j = 1, 2, \ldots, m$, and $w_i, i = 1, 2, \ldots, m$, form a biconjugate A-orthonormal system. The assertion follows immediately from the assumption $\langle w_1, Av_1 \rangle = 1$ in the case $i = j = 1$. Assuming now that $[v_1, v_2, \ldots, v_j]$ and $[w_1, w_2, \ldots, w_j]$ are biconjugate A-orthonormal, we have to show that the augmented sequences $[v_1, v_2, \ldots, v_{j+1}]$ and $[w_1, w_2, \ldots, w_{j+1}]$ are also biconjugate A-orthonormal.

Fact 1. $\langle w_i, Av_{j+1} \rangle = 0$ for $i \le j$.

Fact 2. $\langle w_{j+1}, Av_i \rangle = 0$ for $i \le j$.

We prove below Fact 1. Fact 2 can be obtained in a similar way.

For $i = j$, we have that

$$\begin{aligned} \langle w_j, Av_{j+1} \rangle &= \delta_{j+1}^{-1} \left\langle w_j, A^2 v_j - \beta_j A v_{j-1} - \alpha_j A v_j \right\rangle \\ &= \delta_{j+1}^{-1} \left(\left\langle w_j, A^2 v_j \right\rangle - \beta_j \langle w_j, Av_{j-1} \rangle - \alpha_j \langle w_j, Av_j \rangle \right) \\ &= \delta_{j+1}^{-1} \left(\left\langle w_j, A^2 v_j \right\rangle - \alpha_j \right) = 0 \end{aligned}$$

by the definition of α_j and the induction hypothesis.

For $i < j-1$, we have

$$\begin{aligned}\langle w_i, Av_{j+1}\rangle &= \delta_{j+1}^{-1}\left\langle w_i, A^2v_j - \beta_j Av_{j-1} - \alpha_j Av_j\right\rangle \\ &= \delta_{j+1}^{-1}\left(\left\langle w_i, A^2v_j\right\rangle - \beta_j\langle w_i, Av_{j-1}\rangle - \alpha_j\langle w_i, Av_j\rangle\right) \\ = \delta_{j+1}^{-1}\left\langle w_i, A^2v_j\right\rangle = \delta_{j+1}^{-1}\left\langle A^Tw_i, Av_j\right\rangle &= \delta_{j+1}^{-1}\langle \beta_{i+1}w_{i+1} + \delta_i w_{i-1} + \alpha_i w_i, Av_j\rangle \\ &= \delta_{j+1}^{-1}\left(\beta_{i+1}\langle w_{i+1}, Av_j\rangle + \delta_i\langle w_{i-1}, Av_j\rangle + \alpha_i\langle w_i, Av_j\rangle\right) = 0\end{aligned}$$

by the induction hypothesis.

Finally, for $i = j-1$ we immediately have

$$\begin{aligned}\langle w_{j-1}, Av_{j+1}\rangle &= \delta_{j+1}^{-1}\left\langle w_{j-1}, A^2v_j - \beta_j Av_{j-1} - \alpha_j Av_j\right\rangle \\ &= \delta_{j+1}^{-1}\left(\left\langle w_{j-1}, A^2v_j\right\rangle - \beta_j\langle w_{j-1}, Av_{j-1}\rangle - \alpha_j\langle w_{j-1}, Av_j\rangle\right) \\ &= \delta_{j+1}^{-1}\left(\left\langle w_{j-1}, A^2v_j\right\rangle - \beta_j\right) = \delta_{j+1}^{-1}\left(\left\langle A^Tw_{j-1}, Av_j\right\rangle - \beta_j\right) \\ &= \delta_{j+1}^{-1}\left(\langle \beta_j w_j + \delta_{j-1}w_{j-2} + \alpha_{j-1}w_{j-1}, Av_j\rangle - \beta_j\right) \\ = \delta_{j+1}^{-1}\left(\beta_j\langle w_j, Av_j\rangle + \delta_{j-1}\langle w_{j-2}, Av_j\rangle + \alpha_{j-1}\langle w_{j-1}, Av_j\rangle - \beta_j\right) &= 0\end{aligned}$$

by applying the induction hypothesis.

By construction, we have $\langle w_{j+1}, Av_{j+1}\rangle = 1$, and the proof of the biconjugate A-orthonormalization property of the vector sequences is complete.

Relations (2.4) and (2.5) are matrix reformulations of (2.1)–(2.2).

Relation (2.6) is a matrix reformulation of the first part of the proposition.

Relation (2.7) follows by multiplying both sides of relation (2.4) by W_m^TA and by making use of relation (4.3) and of the associated biconjugate A-orthonormality between W_m and v_{m+1}.

The rest of the proof follows by construction and using relations (2.6)–(2.7). ∎

3. The biconjugate *A*-orthonormalization procedure for solving general linear systems. Using the result of Proposition 1, we develop iterative solutions of system (1.1) by applying an oblique Petrov–Galerkin projection technique onto suitable Krylov subspaces along the following lines.

Step 1. Run Algorithm 1 m steps for some given $m \ll n$ and generate Lanczos-type matrices $V_m = [v_1, v_2, \ldots, v_m]$, $W_m = [w_1, w_2, \ldots, w_m]$ and the tridiagonal projection matrix T_m defined in Proposition 1.

Step 2. Compute the approximate solution x_m that belongs to the Krylov subspace $x_0 + K_m(A; v_1)$ by projecting the following residual orthogonally to the constraints subspace $\mathscr{L}_m \equiv A^TK_m(A^T; w_1)$,

$$r_m = b - Ax_m \perp \mathscr{L}_m.$$

In matrix formulation we may write

$$\left(A^TW_m\right)^T(b - Ax_m) = 0. \tag{3.1}$$

The approximate solution has the form

$$x_m = x_0 + V_m y_m. \tag{3.2}$$

By simple substitution and computation with (2.7), (3.1), and (3.2), we are led to solving the tridiagonal system for y_m,

$$T_m y_m = \beta e_1, \quad \beta = \|r_0\|_2. \tag{3.3}$$

Step 3. Compute the new residual and terminate if convergence is observed. Otherwise, enlarge the Krylov subspace and repeat the whole process.

We are solving implicitly not only the system $Ax = b$ but also the dual linear system $A^T x' = b'$ with A^T. Throughout the paper, we use primed symbols to denote vectors corresponding to the dual system. The dual approximation x'_m is sought from the affine subspace $x'_0 + K_m\left(A^T, w_1\right)$ of dimension m satisfying

$$b' - A^T x'_m \;\perp\; AK_m\left(A, v_1\right).$$

Denote by $r'_0 = b' - A^T x'_0$ the initial shadow residual of the dual system, and put $\beta' = \|r'_0\|_2$. If we choose $w_1 = \frac{r'_0}{\beta'}$ and v_1 such that $\langle v_1, Aw_1\rangle = 1$, the counterpart of the iterative scheme (3.1)–(3.3) for the solution of the dual system $A^T x' = b'$ is written as

$$(AV_m)^T\left(b' - A^T x'_m\right) = 0, \tag{3.4}$$

$$x'_m = x'_0 + W_m y'_m, \tag{3.5}$$

$$T_m^T y'_m = \beta' e_1, \tag{3.6}$$

where V_m, W_m, and T_m are as in Proposition 1 and $y'_m \in \mathbb{R}^n$ is the coefficient vector of the dual linear combination.

Now x_m, respectively, x'_m, can be easily updated from x_{m-1}, respectively, x'_{m-1}. Assume that the LU decomposition of the tridiagonal matrix T_m is

$$T_m = L_m U_m.$$

Substituting this expression into (3.2)–(3.3) and (3.5)–(3.6) gives

$$\begin{aligned}
x_m &= x_0 + V_m\left(L_m U_m\right)^{-1}(\beta e_1)\\
&= x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1)\\
&= x_0 + P_m z_m,\\
x'_m &= x'_0 + W_m\left(U_m^T L_m^T\right)^{-1}(\beta' e_1)\\
&= x'_0 + W_m\left(L_m^T\right)^{-1}\left(U_m^T\right)^{-1}(\beta' e_1)\\
&= x'_0 + P'_m z'_m,
\end{aligned}$$

where $P_m = V_m U_m^{-1}$, $z_m = L_m^{-1}(\beta e_1)$, $P'_m = W_m\left(L_m^T\right)^{-1}$, and $z'_m = \left(U_m^T\right)^{-1}(\beta' e_1)$. Using the same argument as in the derivation of the direct incomplete orthogonalization method (DIOM) from the IOM algorithm in [24, Chapter 6], we easily derive the relations

$$\begin{aligned}
x_m &= x_{m-1} + \zeta_m p_m,\\
x'_m &= x'_{m-1} + \zeta'_m p'_m,
\end{aligned}$$

where ζ_m and ζ'_m are coefficients and p_m and p'_m are the corresponding mth column vectors in P_m and P'_m defined above. We refer to p_m and p'_m as the mth primary and dual direction vectors, respectively.

In the development of our iterative algorithms in the coming sections, we will make extensive use of two properties relating the primary and the dual residual vectors and relating the primary and the dual direction vectors. We state these properties below.

PROPOSITION 2. *The pairs of the primary and dual direction vectors form a biconjugate A^2-orthonormal set, i.e., $\langle p'_i, A^2 p_j \rangle = \delta_{i,j}$ $(1 \le i, j \le m)$.*

Proof. The assertion follows from

$$\begin{aligned}(P'_m)^T A^2 P_m &= \left(W_m \left(L_m^T\right)^{-1}\right)^T A^2 V_m U_m^{-1} \\ &= L_m^{-1} W_m^T A^2 V_m U_m^{-1} \\ &= L_m^{-1} T_m U_m^{-1} \\ &= L_m^{-1} L_m U_m U_m^{-1} \\ &= I_m\end{aligned}$$

with (2.7). ∎

PROPOSITION 3. *The primary and dual residual vectors satisfy the biconjugate A-orthogonal conditions, i.e., $\langle r'_i, A r_j \rangle = 0$ for $i \ne j$.*

Proof. By simple computation with (2.4)–(2.5), (3.2)–(3.3), and (3.5)–(3.6), the mth primary residual vector $r_m = b - Ax_m$ and the mth dual residual vector $r'_m = b' - Ax'_m$ can be expressed as

$$r_m = -\delta_{m+1} e_m^T y_m v_{m+1}, \tag{3.7}$$

$$r'_m = -\beta_{m+1} e_m^T y'_m w_{m+1}. \tag{3.8}$$

Then the assertion follows from (3.7)–(3.8) together with (2.6). ∎

4. The BiCOR method. At this stage the development may proceed in a way similar to the CG method of Hestenes and Stiefel [19] from the coupled two-term vector recurrences with given initial guess x_0,

$$r_0 = b - Ax_0,\ p_0 = r_0, \tag{4.1}$$

$$x_{j+1} = x_j + \alpha_j p_j, \tag{4.2}$$

$$r_{j+1} = r_j - \alpha_j A p_j, \tag{4.3}$$

$$p_{j+1} = r_{j+1} + \beta_j p_j \text{ for } j = 0, 1, \ldots, \tag{4.4}$$

where $r_j = b - Ax_j$ is the jth residual vector and p_j is the jth search direction vector (see, e.g., [31, 33, 5, 29, 28]). In this context, the search direction vectors p_j are multiples of the primary direction. The coupled two-term recurrences for the $(j+1)$th shadow residual vector r'_{j+1} and the associated $(j+1)$th shadow search direction vectors p'_{j+1} can be augmented by relations similar to (4.3)–(4.4) as follows:

$$r'_{j+1} = r'_j - \alpha_j A^T p'_j, \tag{4.5}$$

$$p'_{j+1} = r'_{j+1} + \beta_j p'_j \text{ for } j = 0, 1, \ldots . \tag{4.6}$$

It is important to note that the scalars $\alpha_j, \beta_j,\ j = 0, 1, \ldots$, in the recurrences (4.2)–(4.6) are different from those computed by Algorithm 1. The parameters α_j, β_j are determined by imposing the orthogonality conditions

$$r_{j+1} \perp \mathscr{L}_m \text{ and } A p_{j+1} \perp \mathscr{L}_m. \tag{4.7}$$

The biconjugate A^2-orthonormality between primary and dual direction vectors and the biconjugate A-orthogonality between primary and dual residual vectors stated in Propositions 2 and 3 suggest the adoption of the expanded choice

$\mathscr{L}_m = A^T K_m(A^T, r'_0)$ for the constraints subspace, where $r'_0 = P(A)r_0$ with $P(t)$ an arbitrary polynomial in the variable t. We set $p'_0 = r'_0$. Selecting the optimal polynomial $P(t)$ requires some expertise and artifice. In this study we choose $r'_0 = Ar_0$ for $\mathscr{L}_m$, in contrast with the other common option $r'_0 = r_0$.

We possess now the conditions to determine the scalars α_j and β_j by imposing the corresponding biorthogonality and biconjugacy relations (4.7) into (4.3)–(4.4) and (4.5)–(4.6). Computing the inner product of $A^T r'_j$ and r_{j+1} as defined by (4.3) yields, with the biconjugate A-orthogonality between r_{j+1} and r'_j,

$$\langle A^T r'_j, r_{j+1} \rangle = \langle A^T r'_j, r_j - \alpha_j A p_j \rangle = 0,$$

further resulting in

$$\alpha_j = \frac{\langle A^T r'_j, r_j \rangle}{\langle A^T r'_j, A p_j \rangle},$$

where the denominator can be modified as

$$\langle A^T r'_j, A p_j \rangle = \langle A^T p'_j - \beta_{j-1} A^T p'_{j-1}, A p_j \rangle = \langle A^T p'_j, A p_j \rangle$$

because p'_{j-1} and p_j are A^2-biconjugate. Then

$$\alpha_j = \frac{\langle A^T r'_j, r_j \rangle}{\langle A^T p'_j, A p_j \rangle} = \frac{\langle r'_j, A r_j \rangle}{\langle A^T p'_j, A p_j \rangle}. \tag{4.8}$$

Similarly, using the A^2-biconjugate relation between p_{j+1} as defined by (4.4) and p'_j yields

$$\langle p'_j, A^2 p_{j+1} \rangle = \langle A^T p'_j, A p_{j+1} \rangle = \langle A^T p'_j, A r_{j+1} + \beta_j A p_j \rangle = 0,$$

which gives

$$\beta_j = -\frac{\langle A^T p'_j, A r_{j+1} \rangle}{\langle A^T p'_j, A p_j \rangle} = -\alpha_j \frac{\langle A^T p'_j, A r_{j+1} \rangle}{\langle r'_j, A r_j \rangle}$$

with α_j as computed in (4.8).

Finally, observe that

$$-\alpha_j A^T p'_j = r'_{j+1} - r'_j$$

from (4.5), and therefore

$$\beta_j = \frac{\langle -\alpha_j A^T p'_j, A r_{j+1} \rangle}{\langle r'_j, A r_j \rangle} = \frac{\langle r'_{j+1} - r'_j, A r_{j+1} \rangle}{\langle r'_j, A r_j \rangle} = \frac{\langle r'_{j+1}, A r_{j+1} \rangle}{\langle r'_j, A r_j \rangle} \tag{4.9}$$

because of the biconjugate A-orthogonality of r'_j and r_{j+1}. Combining (4.1)–(4.4), (4.5)–(4.6), and (4.8)–(4.9), and introducing an auxiliary vector recurrence to reduce the number of matrix-vector multiplications, finally leads to the BiCOR algorithm. The pseudocode for the left preconditioned BiCOR method with a preconditioner M is given in Algorithm 2. The structure of the algorithm is partially reflected in the notation. Observe that the vectors fall naturally into paired groups: primal and dual, unpreconditioned and preconditioned, and vector, matrix-vector product pairs. Thus, we denote dual (or shadow) vectors by a primed symbol, e.g., the shadow residual is written r', preconditioned variable names have a prefix z, and a hat symbol $\hat{}$ is used for matrix-vector products. Accordingly, matrix-vector products of preconditioned quantities get a widehat $\widehat{}$ symbol.

ALGORITHM 2. *Left preconditioned BiCOR method.*

Compute $r_0 = b - Ax_0$ for some initial guess x_0.
Choose $r'_0 = P(A)r_0$ such that $\langle r'_0, Ar_0 \rangle \neq 0$, where $P(t)$ is a polynomial in t. (For example, $r'_0 = Ar_0$.)
for $j = 1, 2, \ldots$ **do**
  solve $Mzr_{j-1} = r_{j-1}$
  if j=1 **then**
    solve $M^T zr'_0 = r'_0$
  end if
  $\widehat{zr} = Azr_{j-1}$
  $\rho_{j-1} = \left\langle zr'_{j-1}, \widehat{zr} \right\rangle$
  if $\rho_{j-1} = 0$, **method fails**
  if $j = 1$ **then**
    $p_0 = zr_0$
    $p'_0 = zr'_0$
    $q_0 = \widehat{zr}$
  else
    $\beta_{j-2} = \rho_{j-1} / \rho_{j-2}$
    $p_{j-1} = zr_{j-1} + \beta_{j-2}\, p_{j-2}$
    $p'_{j-1} = zr'_{j-1} + \beta_{j-2}\, p'_{j-2}$
    $q_{j-1} = \widehat{zr} + \beta_{j-2}\, q_{j-2}$
  end if
  $\widehat{q}\,'_{j-1} = A^T p'_{j-1}$
  solve $M^T \widehat{zq}'_{j-1} = \widehat{q}\,'_{j-1}$
  $\alpha_{j-1} = \rho_{j-1} / \langle \widehat{zq}'_{j-1}, q_{j-1} \rangle$
  $x_j = x_{j-1} + \alpha_{j-1}\, p_{j-1}$
  $r_j = r_{j-1} - \alpha_{j-1}\, q_{j-1}$
  $zr'_j = zr'_{j-1} - \alpha_{j-1}\, \widehat{zq}'_{j-1}$
  check convergence; continue if necessary
end for

5. The CORS method: A transpose-free variant of the BiCOR method. Exploiting ideas similar to the ingenious derivation of the CGS method, in this section we present a transpose-free variant of the BiCOR method developed by using a different polynomial representation of the residual with the hope of increasing the effectiveness of BiCOR in certain circumstances. We call the algorithm conjugate A-orthogonal residual squared (CORS).

The derivation of the CORS method is similar to that of the CGS method in [30] and uses the strategy of reducing the number of matrix-vector multiplications by introducing auxiliary vector recurrences and suitable changes of variables. In Algorithm 2, by simple induction, the polynomial representations of the vectors r_j, r'_j, p_j, p'_j at step j can be expressed as

$$r_j = \phi_j(A)r_0, \quad p_j = \pi_j(A)r_0,$$
$$r'_j = \phi_j(A^T)r'_0, \quad p'_j = \pi_j(A^T)r'_0,$$

where ϕ_j and π_j are Lanczos-type polynomials of degree less than or equal to j satisfying $\phi_j(0) = 1$. Substituting these corresponding polynomial representations into (4.8), (4.9) gives

$$\alpha_j = \frac{\langle r'_j, Ar_j \rangle}{\langle A^T p'_j, Ap_j \rangle} = \frac{\langle \varphi_j\left(A^T\right) r'_0, A\varphi_j\left(A\right) r_0 \rangle}{\langle A^T \pi_j\left(A^T\right) r'_0, A\pi_j\left(A\right) r_0 \rangle} = \frac{\langle r'_0, A\varphi_j^2\left(A\right) r_0 \rangle}{\langle r'_0, A^2\pi_j^2\left(A\right) r_0 \rangle},$$
$$\beta_j = \frac{\langle r'_{j+1}, Ar_{j+1} \rangle}{\langle r'_j, Ar_j \rangle} = \frac{\langle \varphi_{j+1}\left(A^T\right) r'_0, A\varphi_{j+1}\left(A\right) r_0 \rangle}{\langle \varphi_j\left(A^T\right) r'_0, A\varphi_j\left(A\right) r_0 \rangle} = \frac{\langle r'_0, A\varphi_{j+1}^2\left(A\right) r_0 \rangle}{\langle r'_0, A\varphi_j^2\left(A\right) r_0 \rangle}.$$

Also, note from (4.3)–(4.4) that ϕ_j and π_j can be expressed by the recurrences

$$\phi_{j+1}(t) = \phi_j(t) - \alpha_j t \pi_j(t),$$
$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j \pi_j(t).$$

By some algebraic computation with the help of the induction relations between ϕ_j and π_j, and using the strategy mentioned above for reducing the number of arithmetic operations, the final algorithm of CORS is obtained. The pseudocode for the left preconditioned CORS method with a preconditioner M can be represented by the scheme illustrated in Algorithm 3. It uses the same notation as Algorithm 2. In many cases, the CORS method is amazingly competitive with the BiCGSTAB method (see, e.g., the numerical experiments of section 6). However, like the CGS method, CORS is based on squaring the residual polynomial. In cases of irregular convergence, this may lead to substantial build-up of rounding errors and worse approximate solutions, or possibly even overflow. For discussions on this effect and its consequences, see, e.g., [24, 32, 27].

6. Numerical experiments. In our numerical experiments, we consider a large set of publicly available linear systems arising from several application areas and having increasing levels of difficulty. We consider both real nonsymmetric and complex non-Hermitian linear systems. Algorithms 1–3 are straightforward to generalize in a complex n-dimensional vector space $\mathbb{C}^n$ using the standard inner product $\langle \cdot, \cdot \rangle$ between two complex vectors $u, v \in \mathbb{C}^n$, defined as

$$\langle u, v \rangle \equiv u^* v = \sum_{i=1}^{n} \bar{u}_i v_i,$$

and using the associated Euclidean vector norm $\|x\| \equiv \sqrt{x^* x}$ and compatible matrix norm (see, e.g., [20]). We summarize in Table 1 the characteristics of the linear systems that were solved. The problem denoted M4D2 arising in computational chemistry is proposed by Sherry Li from NERSC in [1]. The problems denoted STOMMEL1 and STOMMEL2 arising in ocean modeling are proposed by Martin van Gijzen from Delft University in [34]. The other linear systems are extracted from Tim Davis' matrix collection at the University of Florida [7]. The experiments were run in double precision floating point arithmetic in MATLAB 7.7.0 on a PC equipped with an Intel Core2 Duo CPU P8700 running at 2.53GHz and with 4 GB of RAM.

By means of the performance profile tool [10], we evaluate and compare the performance of BiCOR and CORS against other popular iterative algorithms in use today for solving nonsymmetric linear systems. In addition to BiCOR and CORS, we select the following solvers: BiCG, BiCGSTAB, BiCGSTAB(l), CGS, GMRES, QMR, TFQMR. The solvers that we benchmark have slightly different memory requirements, but we do not consider this computational aspect. The aim of our experiments is to show that the two methods presented in this paper can be competitive with other popular approaches in use today for solving nonsymmetric linear systems. In GMRES, the restart parameter which affects the consumed memory is set equal to 50. The memory request for GMRES is the matrix+(restart+3)n, and it remains larger than that needed by BiCOR and CORS (see Table 2). For the BiCGSTAB(l) method, we choose $l = 2$, which shows the overall best performance on most problems in our experiments (we tested $l = 2, \ldots, 6$). No parameter selection is necessary for BiCOR and CORS. In all of our numerical experiments, we choose $r'_0 = Ar_0$. A real implementation would try to recover from possible failures due to this choice.

ALGORITHM 3. *Left preconditioned CORS method.*

Compute $r_0 = b - Ax_0$ for some initial guess x_0.
Choose $r'_0 = P(A)r_0$ such that $\langle r'_0, Ar_0 \rangle \neq 0$, where $P(t)$ is a polynomial in t. (For example, $r'_0 = Ar_0$.)
for $j = 1, 2, \ldots$ **do**
  solve $Mzr_{j-1} = r_{j-1}$
  $\widehat{zr} = Azr_{j-1}$
  $\rho_{j-1} = \langle r'_0, \widehat{zr} \rangle$
  if $\rho_{j-1} = 0$, **method fails**
  if $j = 1$ **then**
    $e_0 = r_0$
    solve $Mze_0 = e_0$
    $d_0 = \widehat{zr}$
    $q_0 = \widehat{zr}$
  else
    $\beta_{j-2} = \rho_{j-1} \,/\, \rho_{j-2}$
    $e_{j-1} = r_{j-1} + \beta_{j-2}\, h_{j-2}$
    $ze_{j-1} = zr_{j-1} + \beta_{j-2}\, f_{j-2}$
    $d_{j-1} = \widehat{zr} + \beta_{j-2}\, g_{j-2}$
    $q_{j-1} = d_{j-1} + \beta_{j-2}\,(g_{j-2} + \beta_{j-2} q_{j-2})$
  end if
  solve $Mzq = q_{j-1}$
  $\widehat{zq} = Azq$
  $\alpha_{j-1} = \rho_{j-1} \,/\, \langle r'_0, \widehat{zq} \rangle$
  $h_{j-1} = e_{j-1} - \alpha_{j-1}\, q_{j-1}$
  $f_{j-1} = ze_{j-1} - \alpha_{j-1}\, zq$
  $g_{j-1} = d_{j-1} - \alpha_{j-1}\, \widehat{zq}$
  $x_j = x_{j-1} + \alpha_{j-1}\,(2ze_{j-1} - \alpha_{j-1} zq)$
  $r_j = r_{j-1} - \alpha_{j-1}\,(2d_{j-1} - \alpha_{j-1}\, \widehat{zq})$
  check convergence; continue if necessary
end for

TABLE 1
Set and characteristics of test matrix problems (listed in increasing matrix size).

Matrix problem	Size	Field	nnz(A)	Row/col diag-dom. scaled A
M4D2	10,000	Quantum mechanics	127,400	2.46% / 5.8%
STOMMEL2	10,491	Ocean modeling	72,965	12.79% / 11.01%
WAVEGUIDE3D	21,036	Electromagnetics	303,468	0.04% / 0.97%
STOMMEL1	42,248	Ocean modeling	294,786	7.57% / 6.71%
WATER_TANK	60,740	3D fluid flow	2,035,281	22.62% / 17.16%
VFEM	93,476	Electromagnetics	1,434,636	0.18% / 0.18%
XENON2	157,464	Materials	3,866,688	0.04% / 0.04%
MAJORBASIS	160,000	Optimization	1,750,416	49.77% / 50.22%
STOMACH	213,360	Electrophysiology	3,021,648	64.58% / 91.73%
TORSO3	259,156	Electrophysiology	4,429,042	86.75% / 95.18%
LANGUAGE	399,130	Natural language processing	1,216,334	81.02% / 78.72%
KIM2	456,976	Complex mesh	11,330,020	0% / 0.58%
ATMOSMODJ	1,270,432	Atmospheric modeling	8,814,880	100% / 100%
ATMOSMODL	1,489,752	Atmospheric modeling	10,319,760	100% / 100%

Our benchmark sent a problem p to each solver s successively, and recorded the solution time $t_{p,s}$ from the start of the solve until either the initial residual was reduced by eight orders of magnitude or the process failed. The process was declared a solver failure when no convergence was achieved after 10,000 matrix-vector products, or the solver stagnated (two consecutive iterates were the same), or when one of the scalar quantities calculated by the algorithm became too small or too large to continue

TABLE 2
Algorithmic and memory cost per iteration for the BiCOR and CORS methods.

Solver	Type	$y = Ax$	$y = A^T x$	$z = ax + y$	$\langle x, y\rangle$	Memory
BiCOR	general	1	1	6	2	matrix+$10n$
CORS	"	2	-	12	2	matrix+$14n$

computing. Solver failures were detected automatically by the MATLAB routines of the iterative algorithms using the default tolerance values. If a solver fails, our script starts solving the next problem. This loop runs through all the problems. For consistency of results, the script repeats each run five times and takes the average.

In our numerical experiments, we did not precondition the linear systems. One reason is because of the memory limits of MATLAB on our computer; the use of preconditioning would require moving to a Fortran implementation. Another reason is that we wanted to study the convergence of the BiCOR and CORS algorithms on matrices with general, possibly tough, distribution of eigenvalues, rather than quickly solving the problems. To reduce the condition number, the linear system was scaled by row and column prior to the iterative solution so that the modulus of each entry of the scaled coefficient matrix is smaller than one. More precisely, we solved $D_1^{1/2} A D_2^{1/2} y = D_1^{1/2} b$, where D_1 and D_2 are the diagonal matrices

$$D_1(i,j) = \begin{cases} \frac{1}{\max\limits_j |a_{ij}|} & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \qquad D_2(i,j) = \begin{cases} \frac{1}{\max\limits_i |a_{ij}|} & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

We used physical right-hand sides b when these were available; otherwise we set $b = Ae$ with $e = [1, \ldots, 1]^T$. The iterative process was always started from $x_0 = 0$, and the same level of convergence tolerance was used for all solvers.

The performance profile shown in Figure 1 is obtained by computing for each problem p the performance ratio of solver s with the best performance by any solver on this problem,

$$r_{p,s} = \frac{t_{p,s}}{\min_s t_{p,s}},$$

and then plotting the cumulative distribution function for the performance ratio

$$\rho_s(\tau) = \frac{1}{n_p} size\{p \ : \ r_{p,s} \leq \tau\}, \ n_p \equiv 14 \text{ (no. of problems)}.$$

The value $\rho_s(\tau)$ gives the probability for solver s to have a performance ratio $r_{p,s}$ within a factor $\tau \in \mathbb{R}$ of the best possible ratio on the set of problems [10]. The value $\rho_s(1)$ is of particular interest because it indicates the probability of solver s being the optimal solver on the set of problems. By comparing this value for all solvers in Figure 1, we see that the CORS algorithm had the highest number of wins in our experiments. The probability that CORS is the optimal solver in our runs is about 0.43. The BiCOR algorithm had less wins, but if we consider a larger region of interest, e.g., to $\tau < 1.5$, its performance is very close to that of CORS.

Observe that the BiCOR method becomes highly competitive in the region of large values of τ. We fix a parameter $r_M \geq r_{p,s}$ for all p, s, so that $\tau \in [1, r_M]$, and we set $r_{p,s} = r_M$ if solver s fails to solve problem p. The probability that the

TABLE 3
Characteristics of the model problems.

Example	Description	Size	Memory (Gb)	Frequency (MHz)	$\kappa_1(A)$
1	Sphere	12,000	4.6	535	$6 \cdot \mathcal{O}(10^5)$
2	Satellite	1699	0.1	57	$1 \cdot \mathcal{O}(10^5)$
3	Jet prototype	7924	2.0	615	$1 \cdot \mathcal{O}(10^7)$
4	Airbus A318 prototype	23,676	18.0	800	$1 \cdot \mathcal{O}(10^7)$

solver solves the problem is given by $\rho_s^* = \lim_{\tau \to r_M} \rho_s(\tau)$, which is the value for which ρ_s flatlines in the graph for large τ. Comparing the quantity ρ_s^* for all solvers, we observe that BiCOR is the solver with the highest probability of success on our problem set. In particular, it is the only solver that converges within the maximum number of iterations on the WAVEGUIDE3D matrix, and it is decidedly the best algorithm on the STOMMEL1 and M4D2 matrices where most of the other methods fail to converge. The BiCOR method failed to solve only one of the 14 test problems, KIM2, while CORS failed to converge on four problems. For the sake of completeness, in Figure 2 we show performance profiles for the number of matrix-vector products. We recall that the iterative algorithms that we benchmark have different costs per iteration; see, e.g., [17]. In Figure 3, we display two examples of convergence histories of CORS and BiCOR representative of the general trend. We clearly see the effect of squaring the residual in the CORS convergence.

Finally, we consider examples where the matrix-vector product is expensive. We select four dense matrix problems, described in Table 3, arising from the method of moment discretization of the following integral equation of the first kind:

(6.1)
$$\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{j}(x) \cdot \vec{j}^t(y) - \frac{1}{k^2} \mathrm{div}_\Gamma \vec{j}(x) \cdot \mathrm{div}_\Gamma \vec{j}^t(y) \right) dxdy = \frac{i}{kZ_0} \int_\Gamma \vec{E}_{inc}(x) \cdot \vec{j}^t(x) dx.$$

Equation (6.1) arises from electromagnetic scattering analysis of perfectly conducting bodies [2]. We denote by $G(|y-x|) = \frac{e^{ik|y-x|}}{4\pi|y-x|}$ the Green's function of the Helmholtz equation, by $\vec{j}$ the unknown current distribution on the surface of the scattering object, by $\vec{j}^t$ the test functions, by Γ the boundary of the object, by k the wave number, and by $Z_0 = \sqrt{\mu_0/\varepsilon_0}$ the characteristic impedance of vacuum (ϵ is the electric permittivity and μ is the magnetic permeability). Using the method of moments, the discretization of (6.1) over a mesh containing n edges leads to dense complex non-Hermitian linear systems $Ax = b$ that are tough to solve by iterative methods. The iteration count of Krylov subspace methods for solving the pertinent linear system typically increases as $\mathcal{O}(n^{0.5})$ [3].

In Table 4, we show the number of iterations required by various Krylov methods to achieve convergence on one node of a Linux cluster equipped with a quad core Intel CPU at 2.8GHz and 16 GB of physical RAM using a Portland Group Fortran 90 compiler version 9. We carried out the matrix-vector product at each iteration using the `ZGEMV` routine available in the LAPACK library, and we did not use preconditioning. The choice of robust preconditioners for this problem class is a challenge in its own right and is beyond of the scope of this paper; we refer the reader to, e.g., [4, 2]. In these experiments we relaxed the stopping criterion used in our runs on sparse problems, due to the slow convergence of Krylov methods and to the $\mathcal{O}(n^2)$ cost of the matrix-vector product. For instance, on Example 1 the CORS method was the only solver to achieve a residual reduction of at least eight orders of magnitude within two hours of CPU time. In the experiments of Table 4 the criterion

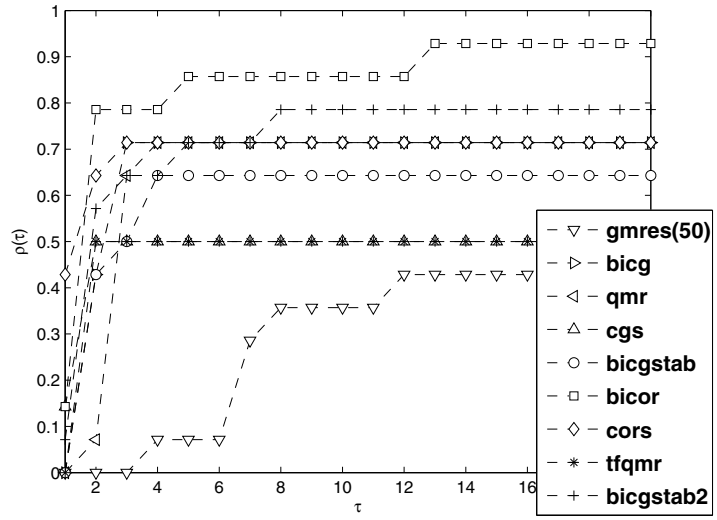


FIG. 1. *Performance profile analysis. The ratio is the CPU computing time.*

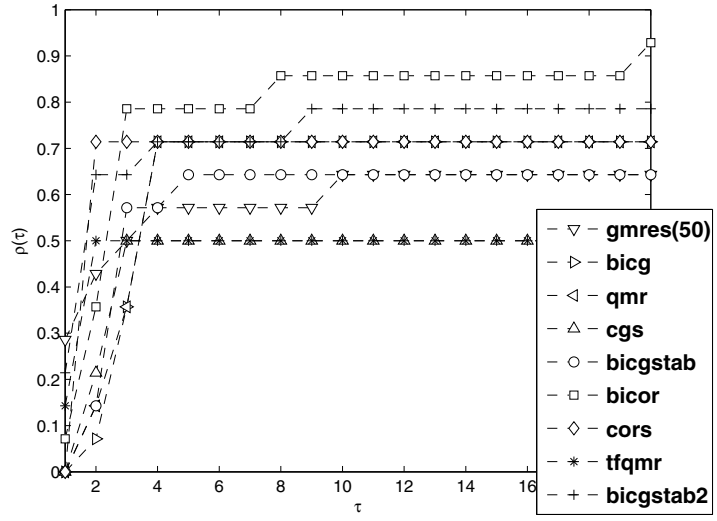
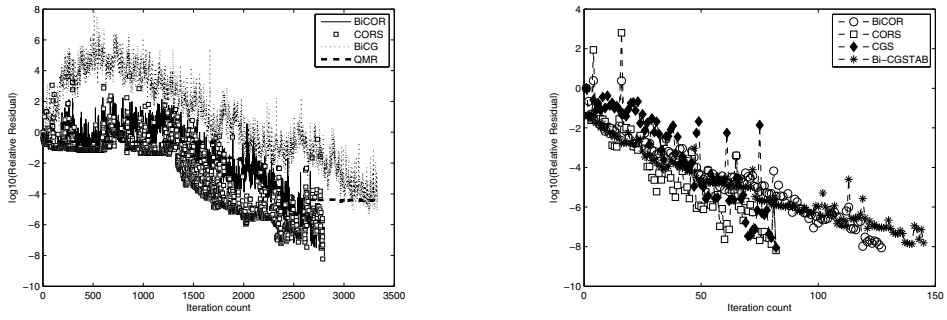


FIG. 2. *Performance profile analysis. The ratio is the number of matrix-vector products.*

for success was reducing the residual norm by a factor of 10^{-5} instead of 10^{-8} as in the first experiments, and the maximum number of matrix-vector products allowed was 3,000 instead of 10,000. The CORS method was the fastest non-Hermitian solver with respect to CPU time on the selected examples. The good efficiency of CORS emerges especially on the two realistic aircraft problems, Examples 3–4. It enabled us to reduce the initial residual to $\mathcal{O}(10^{-3})$ without preconditioning within 3,000 iterates. Below this value of tolerance, none of the solvers converged; hence, we report



(a) On the STOMMEL1 problem. (b) On the TORSO3 problem.

FIG. 3. *Examples of convergence histories.*

only results with the CORS and the restarted GMRES methods in Table 4. This tolerance is considered accurate enough for engineering purposes because it may enable a correct reconstruction of the radar cross section of the object [3]. We also report on the number of iterations and on the CPU time. Another interesting fact about using CORS in this context is that it does not require matrix multiplications by A^H, which often necessitates specific algorithmic implementations for dense matrices.

TABLE 4

Number of iterations and CPU time (in seconds) for experiments on dense matrix problem. Details of the experiments are reported in section 6.

	Example	
Solver	1	2
CORS	380 (**211**)	371 (**11**)
BiCOR	441 (251)	431 (15)
GMRES(50)	> 3000 (> 844)	871 (17)
QMR	615 (508)	452 (24)
TFQMR	399 (435)	373 (27)
BiCGSTAB	764 (418)	566 (18)

	Solver	
Example	CORS	GMRES(50)
3	1286 (**981**)	>3000 (>1147)
4	924 (**54,93**)	2792 (8645)

The large spectrum of experiments on real and complex linear systems reported in this study illustrates the favorable numerical properties of the presented Krylov projection technique. The results indicate that competitive computational techniques may be developed from the biconjugate A-orthonormalization procedure for solving nonsymmetric linear systems.

Acknowledgments. We gratefully thank the EMC Team at CERFACS in Toulouse and the EADS–CCR (European Aeronautic Defence and Space–Company Corporate Research Center) in Toulouse for providing us with some test examples used in the numerical experiments. We would like to thank the anonymous referees for their valuable remarks and comments that greatly improved the quality of the original manuscript.

REFERENCES

[1] M. Baertschy and X. Li, *Solution of a three-body problem in quantum mechanics using sparse linear algebra on parallel computers*, in Proceedings of the 2001 ACM/IEEE conference on Supercomputing (Supercomputing '01), ACM, New York, 2001, CD-ROM, p. 47.

[2] B. Carpentieri, *Algebraic preconditioners for the fast multipole method in electromagnetic scattering analysis from large structures: Trends and problems*, Electron. J. Bound. Elem., 7 (2009), pp. 13–49.

[3] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand, *Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations*, SIAM J. Sci. Comput., 27 (2005), pp. 774–792.

[4] S. H. Christiansen and J.-C. Nédélec, *A preconditioner for the electric field integral equation based on Calderon formulas*, SIAM J. Numer. Anal., 40 (2002), pp. 1100–1135.

[5] M. Clemens, T. Weiland, and U. van Rienen, *Comparison of Krylov-type methods for complex linear systems applied to high-voltage problems*, IEEE Trans. Magnetics, 34 (1998), pp. 3335–3338.

[6] W. M. Coughran, Jr. and R. W. Freund, *Recent advances in Krylov-subspace solvers for linear systems and applications in device simulation*, in Proceedings of the 1997 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD '97), IEEE Press, Piscataway, NJ, 1997, pp. 9–16.

[7] T. Davis, *Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices (2011).

[8] L. G. de Pillis, *A comparison of iterative methods for solving nonsymmetric linear systems*, Acta Appl. Math., 51 (1998), pp. 141–159.

[9] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., 36 (1999), pp. 864–889.

[10] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Programming, Ser. A, 91 (2002), pp. 201–212.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numer. Math. Sci. Comput., Oxford University Press, New York, 2005.

[12] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[13] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, Lecture Notes in Math. 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73–89.

[14] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[15] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[16] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[18] M. H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numer., 6 (1997), pp. 271–397.

[19] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[20] Y.-F. Jing, T.-Z. Huang, Y. Zhang, L. Li, G.-H. Cheng, Z.-G. Ren, Y. Duan, T. Sogabe, and B. Carpentieri, *Lanczos-type variants of the COCR method for complex nonsymmetric linear systems*, J. Comput. Phys., 228 (2009), pp. 6376–6394.

[21] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[22] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[23] B. Parlett, D. Taylor, and Z.-S. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[24] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[25] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[26] G. L. G. Sleijpen and D. R. Fokkema, *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32 (electronic).

[27] G. L. G. Sleijpen and H. A. Van Der Vorst, *An overview of approaches for the stable computation of hybrid BiCG methods*, Appl. Numer. Math., 19 (1995), pp. 235–254.
[28] T. Sogabe, M. Sugihara, and S.-L. Zhang, *An extension of the conjugate residual method to nonsymmetric linear systems*, J. Comput. Appl. Math., 226 (2009), pp. 103–113.
[29] T. Sogabe and S.-L. Zhang, *A COCR method for solving complex symmetric linear systems*, J. Comput. Appl. Math., 199 (2007), pp. 297–303.
[30] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.
[31] E. Stiefel, *Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme*, Comment. Math. Helv., 29 (1955), pp. 157–179.
[32] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.
[33] H. A. van der Vorst and J. B. M. Melissen, *A Petrov-Galerkin type method for solving $Ax = b$, where A is symmetric complex*, IEEE Trans. Magnetics, 26 (1990), pp. 706–708.
[34] M. B. van Gijzen, C. B. Vreugdenhil, and H. Oksuzoglu, *The finite element discretization for stream-function problems on multiply connected domains*, J. Comput. Phys., 140 (1998), pp. 30–46.
[35] A. Zhang and L. Zhang, *Performance of certain Krylov subspace methods for solving convection-diffusion equations*, Appl. Math. Comput., 156 (2004), pp. 695–704.